# A JOINT COMSAT/NBS EXPERIMENT ON TRANSPORT PROTOCOL

D. M. Chitre, H. Y. Chong, and A. Agarwal
COMSAT Laboratories, Clarksburg, MD 20871

R. Colella and K. L. Mills
National Bureau of Standards, Gaithersburg, MD 20899

## ABSTRACT

COMSAT and NBS have embarked on a joint program to examine and test the performance of data communications protocols (specifically, the high level protocols) over satellite links. The first phase of the program focused on the normal data flow procedures of the ISO class 4 transport protocol, which is the lowest level peer-to-peer protocol. This paper discusses the specific procedures in the transport protocol which have an impact on its performance while operating over a satellite link. Modifications to these procedures to improve the performance are described. Next the system configuration and the software implementation for a series of experiments at different bit rates and bit error rates are presented. Finally, the experimental results are discussed and compared with the predicted results obtained by analytical modeling and simulation experiments. Brief descriptions of the analytical and simulation models are also given.

# 1. INTRODUCTION

The International Standards Organization (ISO) has developed a seven-layer reference model for computer communication protocols. Protocols are rules governing the exchange of data in computer networks, and layering of protocols is a result of stratification of functions among parts of the system. The layers are representative of the software and hardware functions needed to perform specific functions. The use of ISO or CCITT specified protocol standards for data communication is increasing rapidly, as it permits the interconnection of systems which are built independently but to the same architectural standard. The communications services to the user or host computer are provided by an explicit network of switching nodes and transmission paths, giving rise to the division between host to network interface and communication subsystems. This results in lower layer protocols called network access protocols and higher layer end-to-end or peer-to-peer protocols.

Figure 1, which shows the ISO reference model, has the lower three layers, physical, link, and network, representing the access protocols. Notice that these access protocols terminate at the first switching node (shown by a three-layer box). Thus, when the access protocols have certain procedures or parameters which have an adverse impact on their operation over a satellite link, appropriate measures can be taken to remove the possible degradation. This is achieved by isolating the access protocol before it reaches the satellite transmission path, e.g., at the first switching node, using an appropriate protocol over a satellite link inside the network and reintroducing the original user access protocol at the destination.

Notice, however, that the higher layer protocols starting with the transport protocol are end-to-end protocols and they are typically host resident. The intervening nodes do not in general have access to these protocols. In many situations the information content pertaining to the higher layers may even be encrypted, making it impossible to intercept them. Thus, it is very important to investigate the higher layer protocols in depth to find out how they would function over a satellite link.

COMSAT and the National Bureau of Standards (NBS) have embarked on a joint program to examine and test the performance of data communications protocols (specifically, the higher layer protocols) over satellite links. Specific objectives of the program are listed below.

a. Demonstrate the successful and efficient operation of the protocol over satellite links for different ranges of transmission speeds and bit error rates.

b. Identify the domain of transmission speeds and bit error rates where the performance degradation takes place.

c. Identify, implement, and test the required modifications in the protocol to remove the degradation of performance.

d. Work towards changing the relevant national and international standards to accommodate the above identified modifications in the protocol.

e. Analyze and simulate the operation of the protocol in the satellite communications environment to help design and gain insight in the joint experiments.

Phase 1 of the program, which was concluded in January 1985 with the experiment over a satellite link, focused on the normal data flow procedures of the ISO class 4 transport protocol [1]. The class 4 transport protocol which is the lowest level peer-to-peer protocol performs, among other functions, error detection and recovery.

There are certain specific procedures in the transport protocol related to acknowledgment and retransmission which have an impact on its performance while operating over a satellite link. Modifications to these procedures to improve the performance were analyzed. These modifications are designed to require only minor changes in the current specification of the class 4 transport protocol, while providing considerable performance improvement over transmission paths with any one or more of the following characteristics:

a. long propagation delay,
b. high bandwidth, and
c. high bit error rate.

The functions of the transport protocol which significantly affect its operation over a satellite link are described in the next section, which also includes a discussion of the appropriate modifications. The third section presents the system configuration of the joint experiment. The software development is described in Section 4. Section 5 describes the sets of experiments and the results and compares them with those obtained by analytical and simulation models.

## 2. CLASS 4 TRANSPORT PROTOCOL

The ISO and the NBS [2] specified class 4 transport protocol (TP-4) perform a range of functions, including connection management, multiplexing and splitting, flow control, transfer of normal and expedited data, and error and loss control. Phase 1 of the COMSAT and NBS joint program dealt only with the normal data transfer and investigated its performance over a satellite link.

One function of the class 4 transport protocol is to detect the loss of a transport protocol data unit (DT TPDU), either due to errors over the transmission paths or due to some other reason e.g., message dropping due to buffer congestion, and to recover the lost or damaged DT TPDU. This is accomplished via an acknowledgment and retransmission mechanism. In the current specification, the transmitted DT TPDUs are sequentially numbered and the sequence number in the acknowledgment message (AK TPDU) from the receiver confirms the reception of all data DT TPDUs with lower sequence numbers. Thus, if there is a gap in the sequence number of received DT TPDUs, the DT TPDUs with sequence numbers beyond the gap cannot be acknowledged. The recovery of missing DT TPDUs is achieved via timer based retransmission. The transmitter retransmits a DT TPDU when the retransmission timer for that DT TPDU expires. These two procedures conspire to produce unnecessary retransmissions and inefficient buffer utilization at the transmitter in many situations, and in particular for a satellite transmission path.

To improve the performance, the acknowledgment mechanism is modified. The modification consists of an additional field in the acknowledgment message (AK TPDU). This field indicates the sequence number of a particular DT TPDU which is being acknowledged. Notice that the acknowledgment message also has the currently specified field with a sequence number confirming all lower sequence DT TPDUs. The transmitter takes appropriate actions based on these two sequence numbers.

## 3. SYSTEM CONFIGURATION

The system configuration shown in Figure 2 was based on a high speed (1.544 Mbit/s) satellite link established between earth terminals located at COMSAT Laboratories in Clarksburg, Maryland, and the NBS in Gaithersburg, Maryland. The space segment was provided by a 14/11-GHz spot beam transponder in an Atlantic Ocean region INTELSAT V F2 satellite. The antennas for the earth terminals consisted of a 2.4-m antenna mounted on a trailer at NBS and a roof-top 4.5-m antenna at COMSAT Laboratories. Four-phase PSK modems with rate 7/8 forward error correction (FEC) were used at each site. A subset of class 4 transport protocol dealing with the normal data flow was implemented in the network interface processor (NIP) system [3] which is a multiprocess/multiprocessor system, tightly coupled via a common bus and a shared memory.

Figure 3 shows the hardware block diagram of the NIP. The processor modules within the NIP are based on commercially available boards using the 32-bit Motorola 68000 microprocessor. Each processor module contains 128 kbytes of memory for program storage and local workspace. Each NIP contains 500 kbytes of high speed memory which is shared among the multiple processors and is used for interprocessor communications. The hybrid multiplexer (HMUX) [3] was used to obtain lower data rate channels (32, 64, and 384 kbit/s) from the 1.544-Mbit/s satellite channel. An in-house developed HDLC module provided a synchronous interface to the hybrid multiplexer (HMUX). The HDLC module only performed HDLC framing, FCS checking, interframe flag generation, zero bit insertion and deletion, and direct memory access (DMA) transfer.

# 4. SOFTWARE DESCRIPTION

The transport protocol was implemented as an application process (or task) under the COMSAT Multiprocessor Operating System (COSMOS). COSMOS is an operating system running on each processor of the NIP. A collection of processors running COSMOS is known as the COSMOS system. The COSMOS system provides the environment in which NIP "application" processes operate. COSMOS contains the basic primitives required for process and resource management, provides facilities for inter-process communications and offers various run time support facilities for application processes.

## 4.1 COSMOS

The COSMOS system consists of various "application" processes running concurrently in a multiprogrammed environment, on various processors of a NIP. All inter-process communications is done using message-passing. A process is statically bound to a specific processor, although it can send messages to any process in the COSMOS system. Except for a tiny amount in the kernel, all software is written in the C programming language.

### 4.1.1 THE COSMOS KERNEL

Every processor in the COSMOS system contains a copy of the COSMOS kernel. The COSMOS kernel provides support for COSMOS processes. It contains the basic scheduler that manages the CPU resource, which is shared by the various processes in a processor. Various flavors of the scheduler exist, which implement different scheduling policies. All are designed to offer very low context switch execution times. The one used in this experiment uses a non-preemptive, non-priority, first-come-first-served scheduling algorithm. Hardware interrupt handling routines are treated as pseudo processes; they are allowed to preempt other processes.

The kernel contains software modules to perform heap management, timer management, process creation/destruction and inter-process communications. Input/Output is not a COSMOS kernel function; instead all I/O is performed by device driver processes. I/O functions are invoked by application processes by sending appropriate messages to device driver processes.

A rich set of packages and utilities to perform various application specific tasks also exist, although they are not considered to be part of the COSMOS kernel.

All COSMOS facilities are invoked as subroutines by application processes; all processes and the kernel run in the same shared address space.

### 4.1.2 INTER-PROCESS MESSAGE COMMUNICATIONS

Inter-process communications in the COSMOS system is done solely by passing messages. (The message system is built on top of a primitive private, binary semaphore facility in the kernel, which is not available to application processes).

Each process has a number of private queues, where messages destined to it arrive. A process can in turn, send messages to a specific queue of any other process in the NIP. Each process has a unique process identifier in the system; messages contain the process identifier of the destination process and the source process. Processes also have character string names; a mapping facility is available in COSMOS to map process names to process identifiers.

Messages are represented as message buffers (also known as Message Control Blocks, MCBs), which are fixed sized memory segments. All messages contain a COSMOS header and a user data area. The COSMOS header is used to route messages to their destinations. The user data area can contain anything, and may further contain other headers for specific application process implemented protocols.

Message passing is implemented by passing pointers to the message buffers; hence, no copying of messages occurs when messages are passed between processes in a NIP. Hence, message passing is not an expensive operation in COSMOS.

### 4.1.3 COSMOS APPLICATION PROCESSES

Most of the functionality of a system comes from the application processes. COSMOS provides the glue from which different systems can be created by developing appropriate application processes. All device drivers are processes (multiple processes in some cases). All communication protocols are implemented as processes. Frequently, different layers of communication protocols are implemented as separate processes. Protocols that support multiple connections are implemented by creating multiple processes, one for each connection; the different processes share one copy of the reentrant program code.

## 4.2 COSMOS CONFIGURATION

Figure 4 shows the inter-relationships of the application processes for this experiment. The traffic generator (TGEN) and SINK processes set up a traffic source/sink configuration for the flow of data messages between the two NIPs.

The USER process allows a user to interactively conduct experiments using the transport protocol. With this process, the user can install a TP process and specify a number of parameters, e.g., TPDU size, window size, timer values, and buffer sizes. The USER process performs the functions of the higher level layers above the transport layer. The HDLCTX process and the HDLCRX process constitute the device driver for the satellite interface hardware (the HDLC hardware module). Note that the link level procedures of the HDLC protocol are not performed by the device driver.

## 4.2.1 TRANSPORT PROTOCOL PROCESS

The TP process implements part of the TP-4 specification. The following services and options were not implemented in this experiment:

- o Graceful Close
- o Transport Connection Multiplexing
- o Expedited Data
- o Flow Control Confirmation

In this Phase 1 experiment, the processes of the two NIPs were configured to perform half-duplex data transfer. The transmit NIP TP process has 256 MCBs allocated as the transmit buffers, while the receive NIP TP process has 256 MCBs allocated as the receive buffers. The retransmission timer value is specified during the TP process installation, and remains fixed.

After a transport connection has been established, a user can request data transfer of specific sizes. The USER process computes the number of DT TPDUs and asks the TGEN process to generate the DT TPDU data portions. The TGEN process then generates the data blocks and sends them to the USER process. After updating the statistics, the USER process forwards the data blocks to the TP process in its user-data queue.

The TP process can be activated by the receipt of a message in its user queue or its network queue or a timer event. It first checks its network queue, and then processes all incoming acknowledgement messages (AK TPDUs). Each AK TPDU has a field called yr_tu_nr which denotes the sequence number of the next expected DT TPDU. For each AK TPDU received, the process cancels the retransmission timers associated with DT TPDUs with sequence numbers up to yr_tu_nr - 1; frees the corresponding transmit TPDU buffers; and restarts the inactivity timer, updates the send window, and sends the DT TPDUs (that have not been sent previously) in the transmit buffers to the HDLCTX process. This continues until the transmit buffers are exhausted or the send window upper edge is reached.

Next, the TP process checks its timer queue. The TP process retransmits a DT TPDU whose timer has already expired. After serving each timer event, the process checks its network queue again and processes any TPDUs waiting there. This continues until the timer queue is empty.

The TP process then checks its user-data queue. If there are any data messages waiting there, the TP process selects the next data message and formats a DT TPDU by calculating the checksum and adding the TPDU header. It then stores the DT TPDU in the transmit buffer. Next, the process checks the send window to determine whether the transmit buffers contain any DT TPDU that has a sequence number within the send window and has not been sent previously. These DT TPDUs are then sent to the HDLCTX process one-by-one until the transmit buffers are exhausted, the send window upper edge is reached, or the maximum number of messages allowed to be queued for the HDLCTX process is reached. The TP process then suspends itself.

In the receive NIP, the HDLCRX process will deliver the TPDUs received over the satellite link to the TP process in its network queue. The TP process then becomes active. It checks its network queue, picks up one message, checks the message type, and calculates the checksum. If a wrong type of TPDU or a bad checksum are discovered, the message is discarded. If the DT TPDU has a sequence number within the receive window, and it is not a duplicate DT TPDU, the DT TPDU is stored in the receive buffer. The TP process checks the receive buffers, retrieves the data portion of TPDUs in sequence, and sends them to the USER process data queue.

Next, the TP process formats an AK TPDU and sends it to the HDLCTX process. The process then returns to its network queue, gets the next message, and processes it in the same way until the network queue is empty or the limit of messages allowed is reached. The TP process serves other queues and then suspends itself and gives up the CPU resource.

## 4.2.2 SELECTIVE ACKNOWLEDGMENT TP PROCESS

The selective acknowledgment TP process implements the modifications in the TP-4 specification to carry out the selective acknowledgment procedure. The only difference between the TP process and this process is in the processing of the AK TPDU.

A selective acknowledgment AK TPDU, has an additional, optional selective acknowledgement sequence number field (sel_tu_nr) not found in the TP-4 AK TPDU (which has a yr_tu_nr field). The yr_tu_nr denotes the next expected DT TPDU sequence number while the sel_tu_nr acknowledges a specific DT TPDU received with sequence number within the receive window. When the transmit NIP's TP process receives an AK TPDU containing the optional sel_tu_nr, it releases the corresponding DT TPDU buffer and cancels the associated retransmission timer.

## 5. EXPERIMENTS

Bulk data transfer experiments were carried out at four rates: 32 kbit/s, 64 kbit/s, 384 kbit/s, and 1,544 kbit/s to test the performance of class 4 transport protocol and the modified selective acknowledgment procedure. Bit error rates in the range of $10^{-8}$ to $10^{-4}$ were obtained by varying the carrier-to-noise ratio. For 32 kbit/s and 64 kbit/s rates, half a megabyte of data was transferred over a transport connection between COMSAT and NBS. Two TPDU sizes were chosen: 128 bytes and 256 bytes. The retransmission timer value was varied between 1 and 1.2 s. For 384 kbit/s and 1,544 kbit/s rates, 2 Mbytes of data were used. The TPDU sizes were chosen as 512 bytes and 1,024 bytes. The retransmission timer value was varied between 0.8 s and 1 s. Also, at these higher rates, two separate cases were considered:

a. the transport protocol performing the checksum calculation, and
b. the transport protocol not performing the checksum calculation.

For each experiment, the following parameters were measured:

a. the number of TPDUs received at the destination,
b. the time interval between the reception of the first TPDU and the last TPDU at the destination,
c. CPU utilization at both ends of the connection,
d. the number of retransmissions,
e. the number of TPDUs which had errors,

The protocol throughput efficiency was calculated from the measured parameter of the number of retransmissions, while the throughput efficiency on the channel was related to the measured time interval between the reception of the first and the last TPDU.

As illustrative examples, Figures 5 and 6 show the improvement in the protocol efficiency when the selective acknowledgment procedure is used. A better than 95-percent efficiency can be achieved over a 1,544-kbit/s satellite channel even at 10-5 bit error rate, using selective acknowledgment procedures. The protocol efficiency was defined to be the inverse of the average number of times a TPDU is transmitted.

These experimental results follow the theoretically predicted results very closely. A model was developed at COMSAT to obtain an analytical expression for protocol efficiency for the class 4 transport protocol with the selective acknowledgement modification. The exact analysis for efficiency of this protocol with a finite receiver buffer size was performed by suitably identifying a Markov state space. The receiver buffer state is defined by two numbers, i and k, where i is the length (in TPDU) of the occupied receiver buffer, including the latest TPDU to arrive, and k is the number of missing TPDUs (gaps) in the buffer. The receiver buffer state is considered only at times when a new TPDU arrives at the receiver for the first time. It can be shown that the arrival of new TPDUs with the above-defined state space follows an aperiodic, nonrecurrent Markov process. The resulting limiting stationary probability distribution (i,k) is computed by solving the Markov equations, where the elements of the transition matrix of the Markov chain are evaluated in terms of error probabilities, retransmission timer, the receiver buffer size, and the round-trip time. Next, these expressions are used to calculate the average number of times a TPDU is transmitted, which in turn gives a measure of protocol throughput efficiency. The analysis closely parallels the Markov analysis carried out earlier [4] for a selective repeat ARQ scheme. When the number of Markov equations to be solved becomes rather large and unmanageable, certain approximate equations obtained by the following analysis can be used. The application of this analysis for TP-4 is illustrated below.

It can be shown that the probability that a DT TPDU will be transmitted n times is given by

$$p(n) = (1 - B^{n+1})^{M-N} (1 - B^{n})^{N} - (1 - B^{n})^{M-N} (1 - B^{n-1})^{N},$$

where B = TPDU error probability,

N = Number of TPDUs which can be transmitted during the time interval between the transmission of a TPDU and the receipt of its acknowledgement,

M = Number of TPDUs which can be transmitted during the time interval between the transmission of a TPDU and the expiration of its retransmission timer.

The average number of times a TPDU is transmitted can then be computed as

$$\bar{n} = \sum_{n=1}^{\infty} np(n)$$

$$= \sum_{k=0}^{M-N} \sum_{\substack{l=0 \\ k+l \neq 0}}^{N} \binom{M-N}{k} \binom{N}{l} (-1)^{k+l} \left[ \frac{B^{k}}{B^{k+l}-1} \right]$$

And the protocol efficiency, $\eta$, is given as

$$\eta = \frac{1}{\bar{n}} .$$

The specific approximations used in the above analysis and their justifications can be found in an earlier paper [5].

A simulation model of the class 4 transport was developed at the NBS [6]. The model is designed to be configurable so that a variety of implementation strategies and environments for the transport protocol can be simulated. The parameters for the simulation experiments were adjusted to reflect as closely as possible the hardware and software configuration of the NIP system. The simulated and live results are compared based on the metric throughput efficiency (TPE), which is defined as the ratio of throughput as viewed by the transport user in bits/second to the link speed, also in bits/second. Less formally, TPE is the fraction of the raw link that the transport user data consumes, excluding protocol overhead due to headers, retransmissions, and link idle time.

Figures 7 through 9 present a comparison between the simulation model and live results for three different experiments. The graphs show good agreement between the predictions of the model and the live data. Figure 7 demonstrates that the transport window mechanism as implemented in the NIP is accurately modeled, both with and without transport checksumming. The performance improvements predicted by the model for selective over TP-4 acknowledgement are verified by Figures 8 and 9 for 64K and 384K bit/second link speeds, respectively.

## 6. CONCLUSIONS

Computer communication via satellite transmission paths can be achieved with satisfactory performance for a wide range of bit rates and bit error rates. The performance degradation of the class 4 transport protocol due to degraded bit error rate and the satellite propagation delay can be alleviated by a minor change in the current specification of the international standard of the protocol.

ACKNOWLEDGMENT

The authors wish to thank all those involved with the planning, implementation, and execution of the joint COMSAT/NBS experiment.

## REFERENCES

[1] International Organization for Standardization, "Transport Protocol Specification," ISO/TC 97/SC 16, International Standard ISO/IS 8073, Rev., June 1984.

[2] National Bureau of Standards, "Specification of a Transport Protocol for Computer Communications, Vol. 3: Class 4 Protocol," February 1983.

[3] A. K. Kaul, et al., "An Experiment in International High-Speed Packet Switching via Satellite," Sixth International Conference On Digital Satellite Communications, Phoenix, Arizona, September 19-23, 1983, Proceedings, pp. II.34-II.43.

[4] D. M. Chitre, "A Selective-Repeat ARQ Scheme and its Throughput Analysis", IEEE International Conference on Communications, June 1982.

[5] D. M. Chitre, "Analysis of the Throughput Efficiency and Delay for ARQ Systems", COMSAT Technical Review, Vol 11, No. 2, Fall 1981.

[6] K. Mills, M. Wheatley, and R. Colella, "Simulation of an International Standard Transport Protocol", 1985 CMG XVI Proceeding, December 1985.

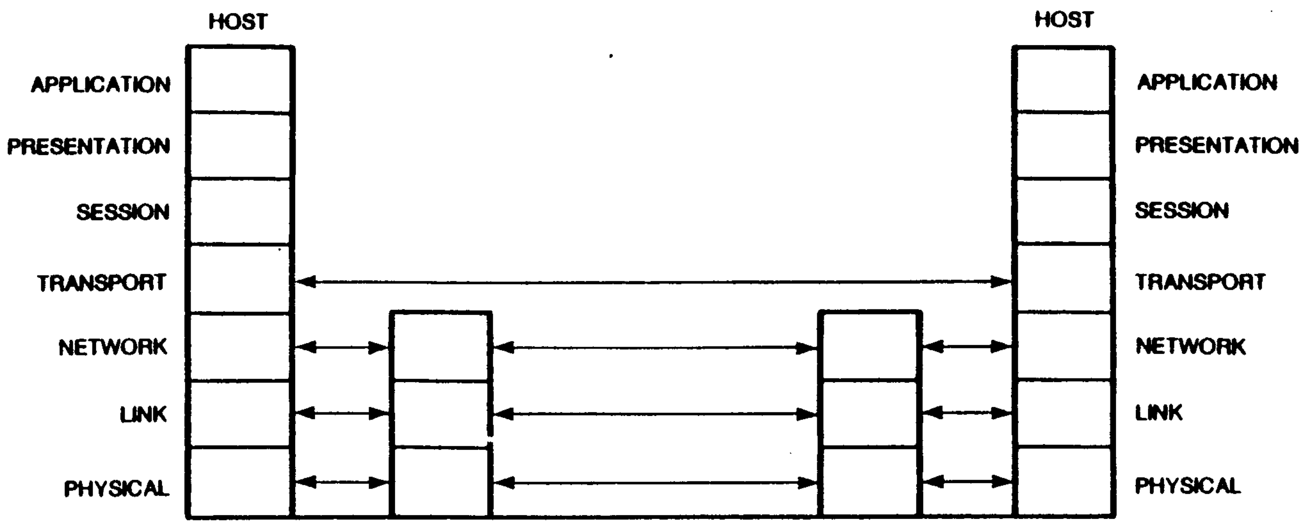
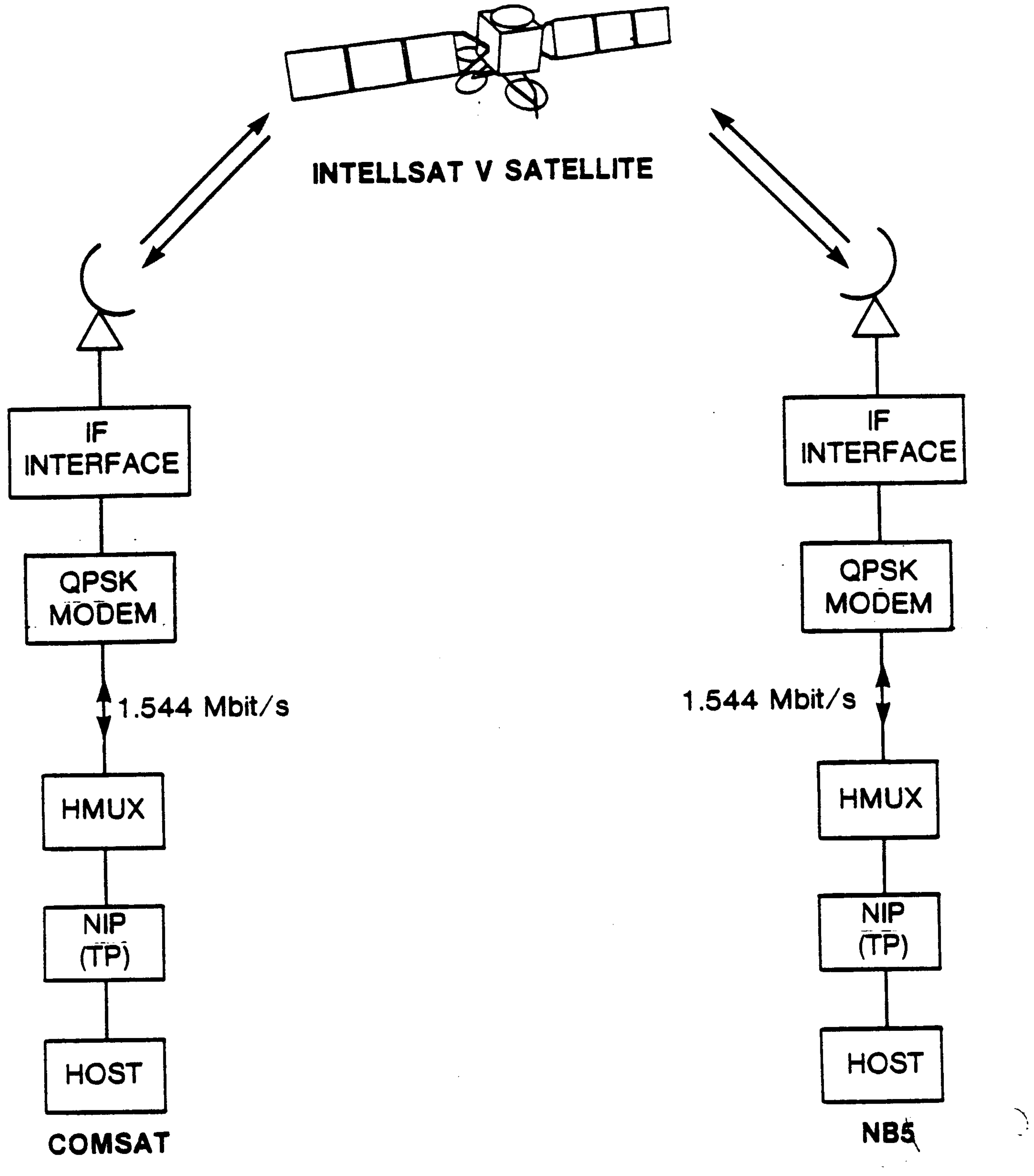


Figure 1. International Standards Organization (ISO) Communications Protocols Reference Model



HMUX - HYBRID MULTIPLEXER

NIP - NETWORK INTERFACE PROCESSOR

TP - TRANSPORT PROTOCOL

Figure 2. System Configuration

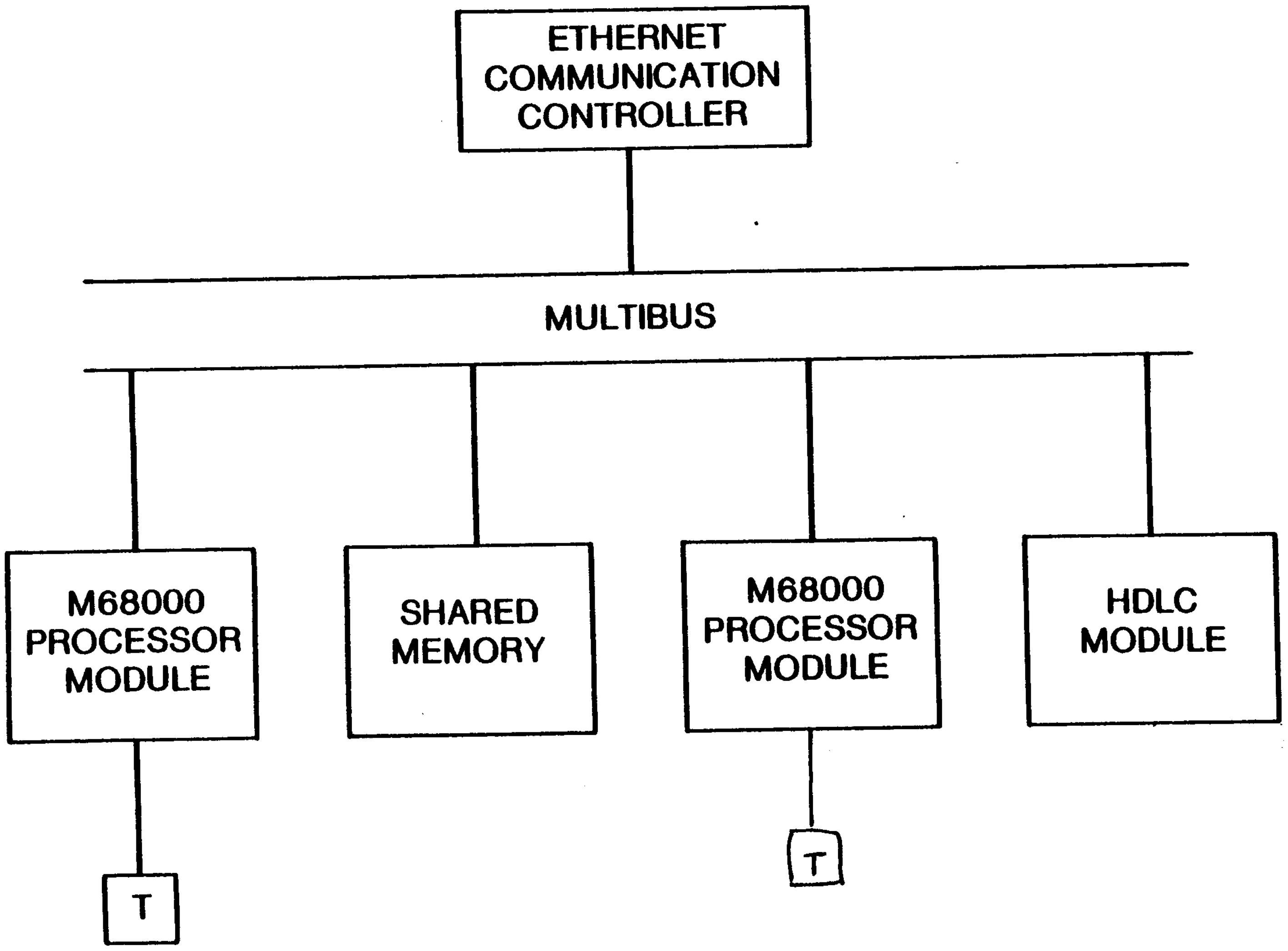


Figure 3. NIP Hardware Block Diagram

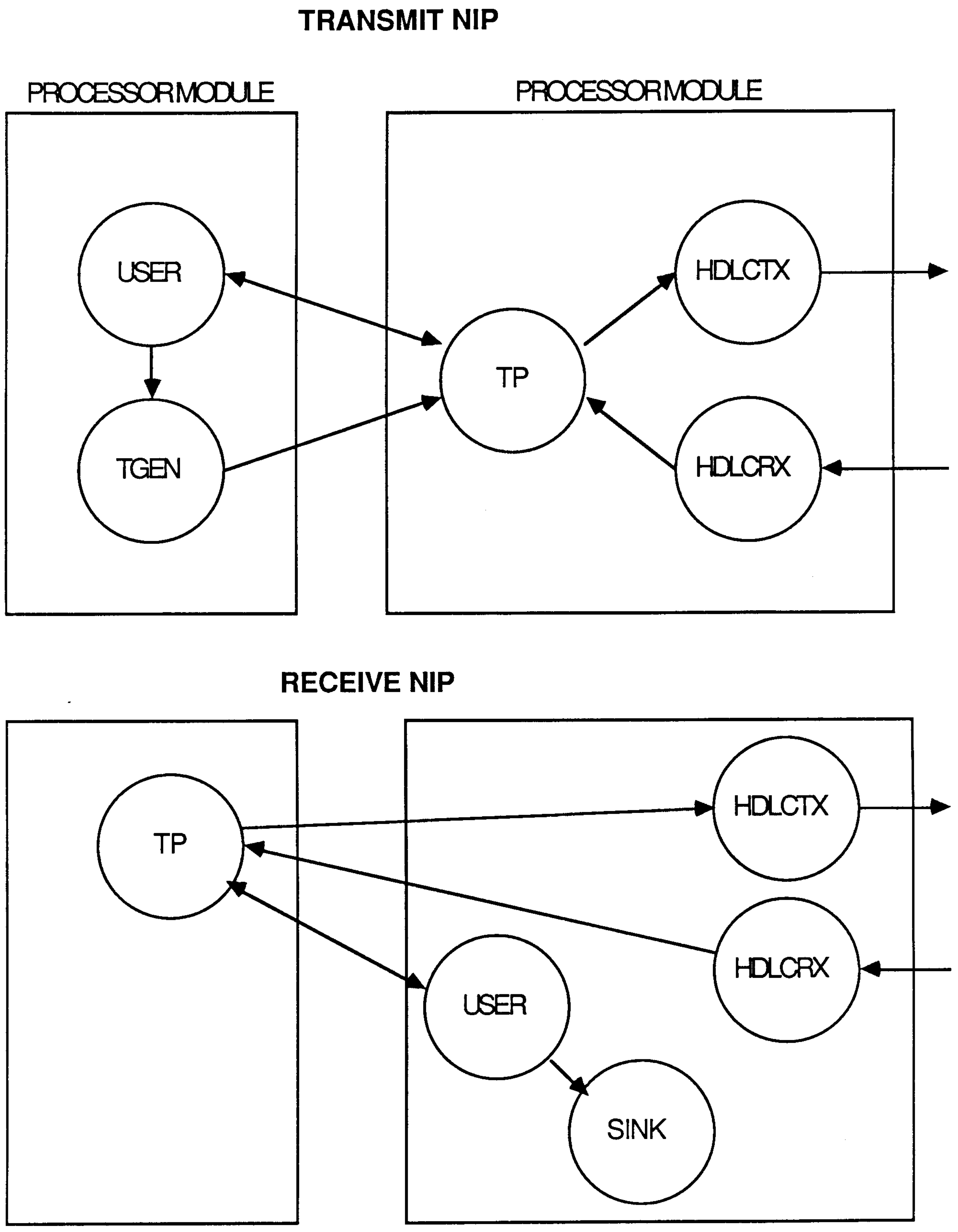


Figure 4. Application Process Configuration

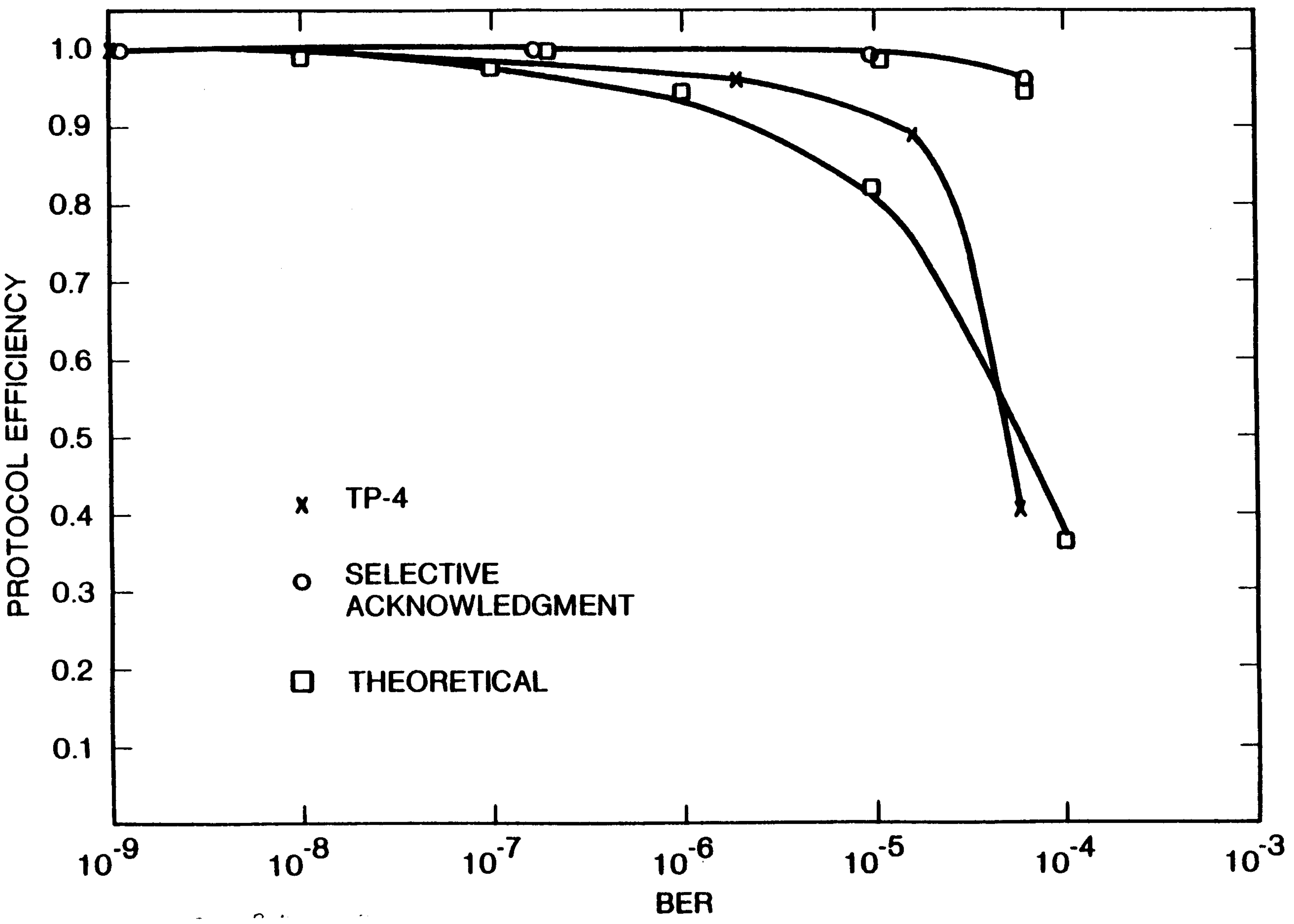
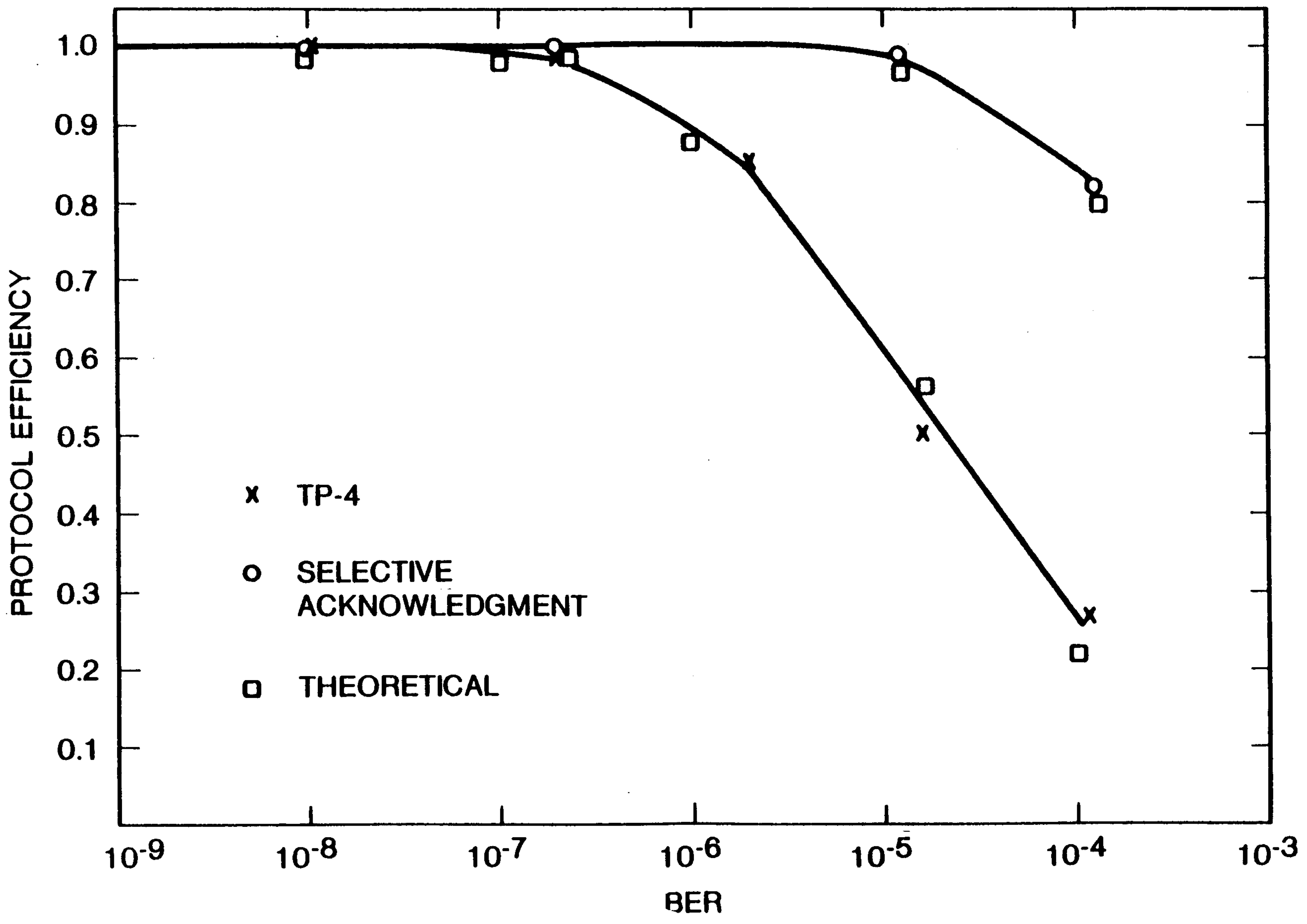


Figure 5. Protocol Efficiency Versus Bit Error Rate for a 64-kbit/s Channel With 128-byte Size Data Unit (Transport Checksumming On)

Figure 6. Protocol Efficiency Versus Bit Error Rate for a 1,544-Mbit/s Channel With 1,024 Byte Size Data Units
(Transport Checksumming Off)



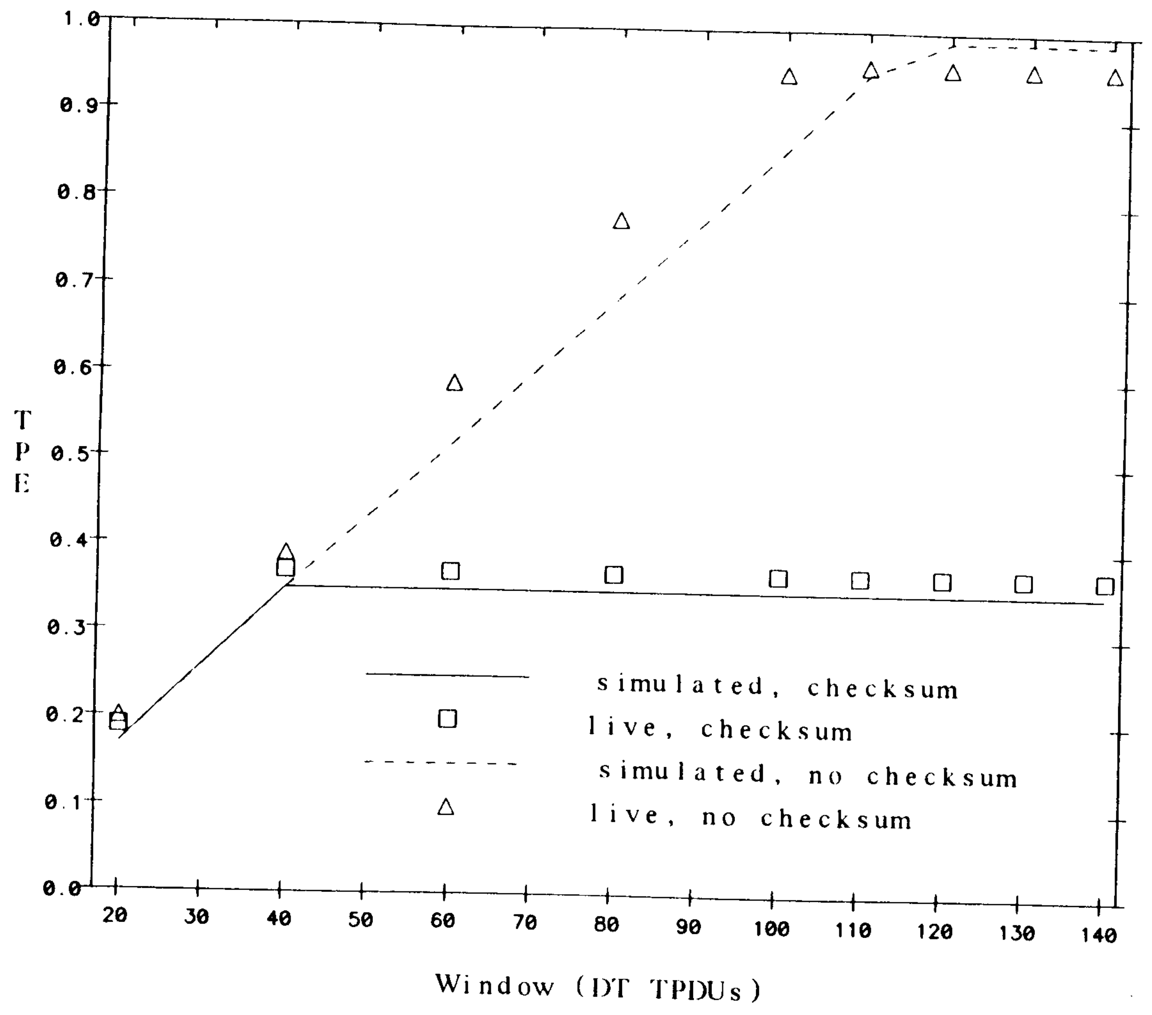


FIGURE 7: TPE as a Function of Window Size, 1.544M bps Link Speed, TP-4

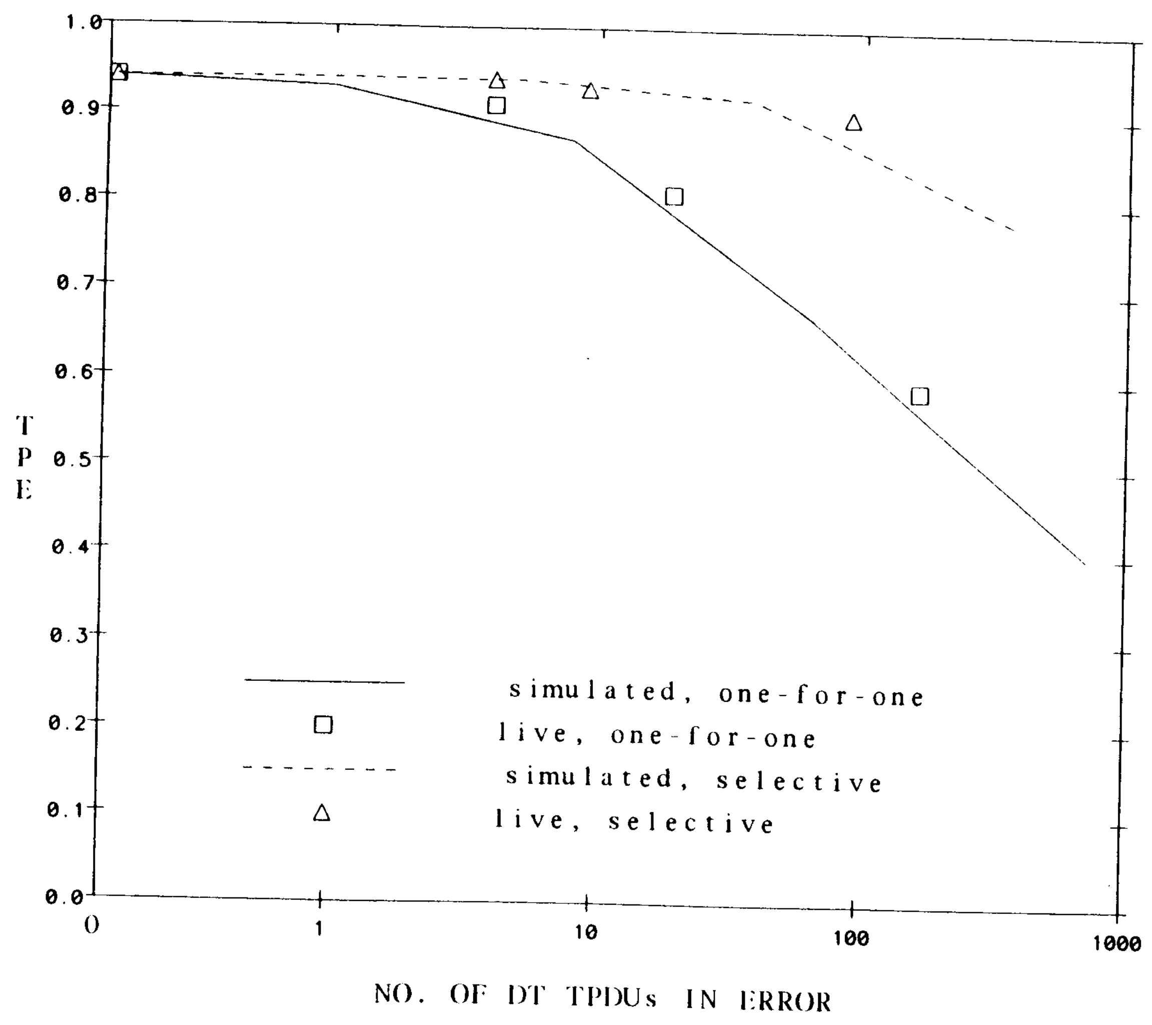


FIGURE 8 : TPE as a Function of No. of DT TPDUs in Error, 64k bps Link Speed, Transport Checksumming On

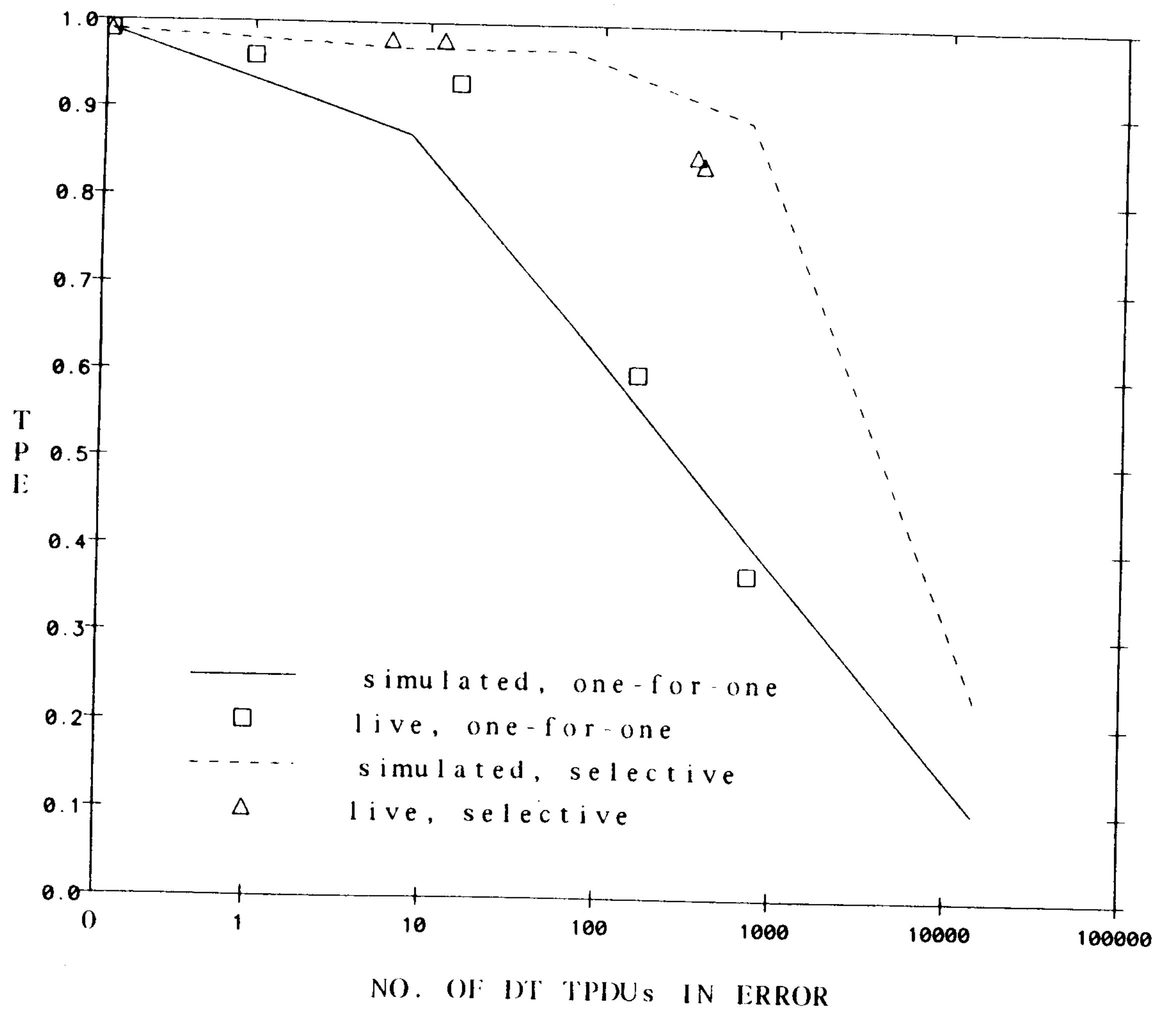


FIGURE 9: TPE as a Function of No. of DT TPDUs in Error, 384K bps Link Speed, Transport Checksumming On